DataEng S24: Data Validation Activity

High quality data is crucial for any data project. This week you'll gain experience with validating a real data set provided by the Oregon Department of Transportation.

Due: this Friday at 10pm PT
Submit: Make a copy of this document and use it to record your results. Store a PDF copy of the document in your git repository along with any needed code before submitting using the in-class activity submission form.

A. [MUST] Initial Discussion Question

Discuss the following question among your working group members at the beginning of the week and place your own response(s) in this space. Or, if you have no such experience with invalid data then indicate this in the space below.

Have you ever worked with a set of data that included errors? Describe the situation, including how you discovered the errors and what you did about them.

Response:
Trisha - no experience
Hector - no experience
Me - no experience
Laura - has had many experiences with data that included errors. One example that she dealt with recently was an issue with user IDs at work where they were no longer present in the data that she was receiving. She had to cross-reference another database which had IDs so that she could properly match up user's information.

Background

The data set for this week is a listing of all Oregon automobile crashes on the Mt. Hood Hwy (Highway 26) during 2019. This data is provided by the Oregon Department of Transportation and is part of a larger data set that is often utilized for studies of roads, traffic and safety.

Here is the available documentation for this data: description of columns, Oregon Crash Data Coding Manual

Data validation is usually an iterative multi-step process.

B. Create assertions about the data
C. Write code to evaluate your assertions.
D. Run the code, analyze the results
E. Write code to transform the data and resolve any validation errors

B. [MUST] Create Assertions

Access the crash data, review the associated documentation of the data (ignore the data itself for now). Based on the documentation, create English language assertions for various properties of the data. No need to be exhaustive. Develop one or two assertions in each of the following categories during your first iteration through the ABC process.

1. *existence* assertions.

   Every crash has a unique Crash ID.

2. *limit* assertions.

   Every crash's speed limit value can't exceed 80 mph.

3. *intra-record* assertions.

   If a crash has a latitude, it must have a corresponding longitude.

4. Create 2+ *inter-record check* assertions.

   The number of vehicle occupants in a crash needs to match the number of injured + uninjured.

   Each crash record with more than 0 vehicles involved, needs to have more than 0 participants involved.

5. Create 2+ *summary* assertions.

   There were thousands of crashed but not millions

   The number of fatal crashes is much less than non-fatal crashes.

6. Create 2+ *statistical distribution assertions*.

Crashes are more frequent on the weekends.

Crashes peak during holiday seasons.

These are just examples. You may use these examples, but you should also create new ones of your own.

C. [MUST] Validate the Assertions

1. Study the data in an editor or browser. Study it carefully, this data set is non-intuitive!.
2. Write python code to read in the test data. You are free to write your code any way you like, but we suggest that you use pandas' methods for reading csv files into a pandas Dataframe.
3. Write python code to validate each of the assertions that you created in part A. The pandas package eases the task of creating data validation code.
4. If needed, update your assertions or create new assertions based on your analysis of the data.

D. [MUST] Run Your Code and Analyze the Results

In this space, list any assertion violations that you encountered:

- AssertionError: Occupants must equal the sum of injured and uninjured.
    - There needs to be many columns added into this calculation I believe.
- AssertionError: Every crash must have a unique Crash ID. (CHANGED THIS ONE)
    - There are multiple entries for each crash, I'm not sure how to distinguish, so I changed this to just check that every row has a Crash ID
- AssertionError: There are more crashes with injuries than crashes with none.
    - More crashes involve no injuries, change assertion
- AssertionError: More crashes should occur on Saturday than on Sunday.
    - Assert that more crashes occurred on Sunday.

For each assertion violation, describe how to resolve the violation. Options might include:

- revise assumptions/assertions
- discard the violating row(s)
- Ignore
- add missing values
- Interpolate
- use defaults
- abandon the project because the data has too many problems and is unusable

No need to write code to resolve the violations at this point, you will do that in step E.

E. [SHOULD] Resolve the Violations and Transform the Data

For each assertion violation write python code to resolve the violation according to your entry in the “how to resolve” section above.

Output the validated/transformed data to new files. There is no need to keep the same, awkward, single file format for the data. Consider outputting three files containing information about (respectively) crashes, vehicles and participants.

F. [ASPIRE] Learn and Iterate

The process of validating data usually gives us a better understanding of any data set. What have you learned about the data set that you did not know at the beginning of the current ABC iteration?

Next, iterate through the process again by going back through steps B, C, D and E at least one more time.